IN THE SPECIFICATION:

On page 1, lines 4-10, replace the entire paragraph with the following:

This application is a continuation of U.S. patent application 10/364,147, filed February 11, 2003, now U.S. Patent 6,711,041, which is a continuation of U.S. patent application 09/940,832, filed August 27, 2001, now U.S. Patent 6,542,391, which is a continuation-in-part of each of the following U.S. patent applications: application no. 09/590,642, filed June 8, 2000; application no. 09/590,428 filed June 8, 2000; application no. 09/590,775 filed June 8, 2000; application no. 09/594,206, filed June 14, 2000; application no. 09/594,209, filed June 14, 2000; application no. 09/594,201, filed June 14, 2000; application no. 09/594,194, filed June 14, 2000; and application no. 09/594,202, filed June 14, 2000. Each of the above-identified applications are hereby incorporated by reference in their entirety.

On page 10, lines 21-23, replace the entire paragraph with the following:

Figure 75 illustrates an exemplary operation of the instruction decoder of Figure 59 in response to an instruction to read a CAM word from the highest priority match address of a class-based partition of the CAM array;

On page 10, line 26, replace the entire paragraph with the following:

Figure 77 depicts a CAM block with two classes of data stored therein.

On page 14, line 31-page 15, line 4, replace the entire paragraph with the following:

Figure 2 shows a CAM array 200 that is one embodiment of a CAM array 104 of Figure 1. The array 200 includes a plurality of CAM cells 202 organized in any number of rows and columns. Each row of CAM cells 202 is coupled to a match line ML and a word line WL. Each word line WL is driven by an address decoder 204 to select one or more of CAM cells 202 for writing or reading. For alternative embodiments, multiple CAM blocks may share a decoder. Each match line ML provides the match results of a compare operation to the priority encoder 108 (see also Figure 1). A match line ML indicates a match condition for the row only if all CAM cells 202 in that row match the comparand data. Each CAM cell 202 may be a binary, ternary, SRAM-based or DRAM-based CAM cell. In some embodiments, the match line ML is pre-charged for the compare operation. If any CAM cell 202 in the row does not match the comparand data, the CAM cell(s) 202 discharges the match line ML toward ground potential (e.g., logic low). Conversely, if all CAM cells 202 match the comparand data, the match line ML remains in a charged state (e.g., logic high). When the CAM block 102 is disabled in response to the select signal SEL, the comparand word is not driven into the array 200, and the match lines ML may remain in their charged state during the compare operation, regardless ~~if~~of whether there is a mismatch. The match lines need not be pre-charged for a subsequent compare operation. The ability to maintain the match lines of unselected CAM blocks in their charged state during the compare operation may further reduce power consumption of present embodiments over prior art architectures.

On page 20, lines 1-10, replace the entire paragraph with the following:

During a compare operation, each CAM block 702 receives comparand data from the comparand bus CBUS in a manner similar to that of CAM blocks 102 of device 100 of Figure 1. Other signals provided to device 700 during the compare operation may be a clock signal CLK, one or more instructions from an instruction decoder (not shown for simplicity), and other control signals. Each CAM block 702 provides a plurality of match line signals to the priority encoder 708 via corresponding match lines ML. The match lines carry match signals indicative of match conditions in the CAM arrays 704. For simplicity, the plurality of match lines ML from each CAM block 702 are represented collectively in Figure 7. The priority encoder 708 generates an index corresponding to one of the matching CAM words in the device 700, which as described above may be the index of the highest-priority matching CAM row.

On page 20, lines 11-23, replace the entire paragraph with the following:

Each CAM block 702 provides a full flag signal FF indicative of whether the CAM block is full, i.e., whether there are any available rows~~row~~ in the CAM block 702 to store data, to full flag logic 710. The full flag signal FF may be generated for each CAM block 702 in a well-known manner using one or more valid bits in each row of the CAM block. The full flag signals FF_1 to FF_n provided by CAM blocks 702(1)-702(n), respectively, are combined in a well-known manner in full flag logic 710 to generate a device full flag signal, FF_device, indicative of whether there are any available rows in the device 700. When a CAM block 702 is found to be defective or otherwise inoperable for its intended purpose, the CAM block 702 is configured to maintain an asserted full flag signal FF to indicate that the defective CAM block 702 does not include any available memory locations. In one embodiment, the full flag signal FF for the defective CAM block may be maintained in the asserted state by forcing the valid bits in its array 704 to an asserted state. In other embodiments, a fuse may be provided within or associated with each CAM block 702 that, when blown, forces the corresponding full flag signal FF to be asserted.

On page 24, line 26-page 25, line 10, replace the entire paragraph with the following:

The CAM device 800 is then tested to determine if any CAM blocks are defective. Where it is determined that one or more CAM blocks are defective, the select values stored in memory elements 1006(0)-1006(3) may be modified to re-address the non-defective CAM blocks. For example, if after testing it is determined that CAM block 802(0) is defective, and is thereafter disabled using the corresponding block select circuit 706 as described above (see also Figure 7), the 1k CAM rows in the defective CAM block 802(0) are no longer available, and therefore the device 800 now has only 3k available CAM rows available, i.e., 1k rows in each of the 3 non-defective CAM blocks 802(1)-802(3). Since the first CAM block 802(0) is not available, it is desirable for the second CAM block 802(1) to be the highest-priority CAM block (e.g., having address space 0 to k-1), for the third CAM block 802(2) to be the second highest-priority CAM block (e.g., having address space k to 2k-1), and for the fourth CAM block 802(3) to be the third highest-priority CAM block (e.g., having address space 2k to 3k-1).

On page 30, lines 4-13, replace the entire paragraph with the following:

The concatenation of a 12-bit row index I and the 2-bit block index forms a 14-bit device index of the highest-priority match, if any, from a corresponding CAM block 802. The 1 input of the first multiplexer 1202(0) receives~~receive~~ a default binary "0" value. MF_0 is inverted by inverter 1206 and provided as the select signal to multiplexer 1202(0), and provided directly as the select signal to multiplexer 1202(1). MF_0 and MF_1 are combined in OR gate 1208 and provided as the select signal for multiplexer 1202(2). MF_2 and the result from OR gate 1208 (i.e., MF_0 + MF_1, where + is the logic OR function) are combined in OR gate 1210 and provided as the select signal for multiplexer 1202(3). As explained below, the match flags MF control whether each multiplexer 1202 passes a concatenated device index from a previous CAM block or the concatenated device index of the corresponding CAM block.

On page 30, lines 14-19, replace the entire paragraph with the following:

In this example, CAM block 802(0) is the highest-priority block, CAM block 802(1) is the next highest-priority block, and so on. For each multiplexer stage, if there is a match in the corresponding CAM block 802, the row index I and block index are forwarded to the next stage if there is not a match condition in a previous or higher-priority CAM block 802. If there is a match condition in a higher-priority CAM block 802, the row index I plus the block index from the higher-priority CAM block are forwarded to the next stage.

On page 35, lines 17-24, replace the entire paragraph with the following:

The input address on the address bus has twelve bits A11-A0. In the 4k x 72 mode, all twelve bits A11-A0 are used to uniquely address each of the 4k row segments in CAM array 1102. Bits A11-A2 are selected by RA select logic 1283 and are used as the row address for row decoder 1282 to select one of the ~~CAM rows;~~ CAM rows, and bits A1-A0 are provided to SA select logic 1286 and used to select one of the row segments for a selected row of cells. In this mode, SZ72 is enabled and SA select logic 1286 provides A1 and A0 as SD1 and SD0, respectively, to segment decoder 1284. A1 and A0 are decoded by segment decoder 1284 to generate SEN1-SEN4 and select a particular row segment in a selected row of cells for communication.

On page 37, lines 7-16, replace the entire paragraph with the following:

Figure 15 illustrates~~disclosed~~ a particular example of the operation of decoder 1280 for a particular number of possible CAM array configurations. The method used in the example of Figure 15 can be readily extended to accommodate any number of configurations of any size CAM array having any number of row segments each having any number of CAM cells. For example, a CAM array having more row segments can be accommodated by supplying more address bits (SA), select signals, and configuration signals to SA select logic 1286 (and/or RA select logic 1283), and increasing the number of SDA bits, the size of segment decoder 1284 and the number of segment enable signals. In general, the row address will have $\log_2 Y$ bits to select one of the Y word lines, and the SA address, SSEL and SDA will each have up to $\log_2 Z$ bits to address one of the Z segment enable lines.

On page 39, lines 13-19, replace the entire paragraph with the following:

In the 1k x 288 mode, ten bits A11-A2 are used to uniquely address each of the 1k groups of row segments in CAM array 1102. Each group of row segments includes four row segments (i.e., an entire row). The most significant ten bits A11-A2 are used as the row address for row decoder 1702 to select one of the CAM rows. In response to SZ288, multiplexer 1706 provides ~~and~~ the ~~all~~ logic one states of input port IP3 to SEN1-SEN4. This enables an entire selected row to simultaneously communicate with the data bus. The least significant bits A1 and A0 do not participate in addressing a group of row segments.

On page 48, lines 12-29, replace the entire paragraph with the following:

Figure 39 shows row match circuit 3700 that is one embodiment of row multiple match circuit 3604(0) of Figure 38. Row match circuit 3700 may be used for each row multiple match circuit. Row multiple match circuit 3700 includes multiple match one logic 3702, group multiple match logic circuits 3704(1)-3704(n-2), and multiple match configuration logic 3706. Multiple match one logic 3702 determines a multiple match condition in row 1122(0) for the ZY x W mode. Multiple match one logic 3702 receives each of the match line segments M1-MZ from row segments S1-SZ, respectively, and generates MMONE indicative of whether more than one row segment stores data that matches the comparand data. That is, multiple match one logic 3702 determines when two or more of M1-MZ are enabled. When configuration signal SZ1 is enabled, match configuration logic 3706 outputs MMONE as the row multiple match signal MMR0. Any multiple match logic circuitry can be used for logic 3702 to determine a multiple match condition. One embodiment of multiple match one logic is shown in Figure 40. For this embodiment, ~~r two-input r~~ two-input AND gates 3802(1)-3802(r) each receive a unique combination of two of the match line segments, where r is determined by the combinatorial formula r=Z!/(2!(Z-2)!). The output of each AND gate is provided to OR gate 3804 to generate MMONES. One example of the approach of Figure 40 for four row segments is shown in Figure 41, where all of the combinations of the four match lines segments taken two at a time are provided to AND gates 3902(1)-3902(6), and the outputs of the AND gates are provided to OR gate 3904.

On page 52, lines 20-27, replace the entire paragraph with the following:

Figure 51 shows logic 4900 that is one embodiment of generating PSA0(1). Other embodiments~~embodiment~~ may be used. Logic 4900 includes NAND gates 4902, 4904, 4906, and 4910, and NOR gate 4908. NAND gate 4902 has one input coupled to M1 and another input coupled to M2. NAND gate 4904 has one input coupled to the output of NAND gate 4904 and another input coupled to SZ144. NAND gate 4906 has one input coupled to the output of NAND gate 4906 and the other input coupled to the output of NAND gate 4910. NAND gate 4910 has one input coupled to SZ72 and the other input coupled to the output of NOR gate 4908. NOR gate 4908 has one input coupled to M1 and the other input coupled to M2.

On page 61, lines 5-11, replace the entire paragraph with the following:

An operation select signal 6208, preferably generated by the instruction decoder, and the block configuration signal 5903 are input to each row flag circuit 6202. As discussed below, the operation select signal 6208 is used to select which set of input signals (i.e., match signals M_1-M_Z or validity signals V_1-V_Z) are to be operated upon by logic circuits within the row flag circuit 6202. In one embodiment, the block configuration signal 5903 is used to select one or more of the logic circuits to output a row flag signal 6207 from the row flag circuit according to width/depth configuration of the CAM block.

On page 66, lines 8-15, replace the entire paragraph with the following:

Figure 67 illustrates an embodiment of the global priority encoder 5918 of Figure 59. A multiplexer 6702 having ports P0-P(n-1) receives a select signal 6705 from a block index select circuit 6704 that selects one of the block indexes~~indices~~ 5917(0)-5917(n-1) and a corresponding block identifier 6707(0)-6707(n-1) to be output as the device index 5920. As discussed above, the block index is alternatively indicative of a match address or a not-full address (also called a free address) according to whether a compare or write operation is being performed. Accordingly, the device index indicates a highest priority match address during a compare operation and indicates a highest priority not-full address (i.e., a "next free address") during a write operation.

On page 69, line 30-page 70, line 12, replace the entire paragraph with the following:

As mentioned above in reference to Figure 59, the address circuit 5906 may include register banks for maintaining class-based addresses to access the CAM array in response to certain read and write instructions. For example, in one embodiment, the address circuit 5906 maintains a bank of highest-priority-match (HPM) registers to store the device indexes~~indices~~ that result from class-based compare operations. For example, if the blocks of the CAM array are partitioned into three classes of storage, A, B and C, then the device index that results from a compare operation in the class A storage (i.e., a class A compare) may be stored in a register of the HPM register bank that corresponds to class A. Similarly, the device index that results from a class B compare may be stored in a register of the HPM register bank that corresponds to class B and a device index that results from a class C compare may be stored in a register of the HPM register bank that corresponds to class C. By this arrangement, read operations which reference the highest priority match addresses on a class basis may be supported. For example, if a host processor performs a sequence of compare operations at classes A, B and C, then desires to read the contents of the highest priority match address for a given class, the host processor may issue a read instruction referencing the HPM register for the class (referred to as a READ@HPM@CLASS instruction). The appropriate HPM register within the HPM register bank will then be selected to provide the address for the read operation.

On page 72, line 15-page 73, line 3, replace the entire paragraph with the following:

As shown, ~~the~~ when the control signal is deasserted, no register load signal is asserted. Thus, when an incoming instruction specifies an operation (e.g., a read operation) that does not produce a device index, the instruction decoder may deassert the control signal to prevent the HPM and NFA register banks from being loaded. Also, no register load signal is asserted when the device flag signal is deasserted. Recalling that the device flag signal is asserted when a match is detected during a compare operation or when a storage partition includes at least one unfilled storage location after a write operation, a non-asserted device flag conversely indicates that no match was found in the compare operation or that the storage partition is full after the write operation. In either event, the device index does not represent a valid address within the CAM array (i.e., neither a match address nor a not-full address) when the device flag is deasserted. Accordingly, no register load signal is asserted when the device flag is not asserted.

On page 74, lines 13-20, replace the entire paragraph with the following:

Figure 75 illustrates an exemplary operation of the instruction decoder 5904 of Figure 59 in response to an instruction to read a CAM word from the highest priority match address of a class-based partition of the CAM array (i.e., a READ@HPM@CLASS instruction). In block 7501, the instruction decoder issues the appropriate select and class code signals to the address circuit (e.g., element 5906 of Figure 59) to select the HPM register for the specified class code to source the address for a read access to the CAM array. A first predetermined time later, in block 7503, the instruction decoder signals a read circuit within the CAM device (not shown in Figure 59) to sense data output from the CAM array location selected by the address circuit.